Common Ground

A PUBLICATION OF INTEL CORPORATION AUTOMOTIVE OPERATION VOLUME 2, NO. 1, 1993

CONTENTS

WHAT IS FUZZY LOGIC?

By Rob Kowalczyk
Applications Engineer, Intel Corporation

Fuzzy Logic, based on the work of Lotfi Zadeh, offers a novel approach to implementing control systems. The Japanese and Europeans have been working with fuzzy logic for many years. They have developed fuzzy control systems for everything from subways to washing machines. While the term "fuzzy logic" on products is often present for its marketing value, fuzzy logic does offer a new way to look at developing control systems.

Fuzzy logic defined

Fuzzy logic is a misnomer. There is nothing fuzzy about it. A better name would be multi-valued logic. Fuzzy logic has its basis in fuzzy set theory developed by L. A. Zadeh of the University of California at Berkeley. This is a well-understood (although not widely studied) mathematical discipline. Fuzzy set theory is as rigorously defined as conventional set theory or conventional control theory. It defines operations that can be performed on fuzzy sets, such as union, intersection and inclusion, and clearly states how these operations are performed. Furthermore, performing a given operation on a set or group of sets will always yield a predictable, consistent result.

Fuzzy vs conventional set theory

In conventional set theory an object either is or is not a member of a set. Fuzzy sets, on the other hand, allow objects to have a continuous degree of membership (i.e., a number between 0 and 1). For example the number "3" is a member of the set of odd numbers, but is not in the set of even numbers. This binary mapping is a characteristic of conventional sets, but is not adequate for all sets.

Consider the statement "it is very hot outside." It is not possible to determine one exact temperature above which everybody would agree that it is very hot. Even one person may not be able to determine one temperature above which he or she would absolutely say it is very hot.

Figure 1 shows example membership functions for both fuzzy and traditional sets. Note that for the traditional set, an item is either a member of the set or it is not. The degree of membership is either 0 or 1. However, in the case of fuzzy sets, an item can possess a degree of membership with any value between 0 and 1, inclusive. Thus for a temperature of 90°F using the membership function for traditional sets, we see that 85°F is very hot. However, somebody living in Arizona might disagree. Using the membership function given in Figure 1 for fuzzy sets, we could say that a temperature of 85°F is very hot with a degree of membership of 0.8.

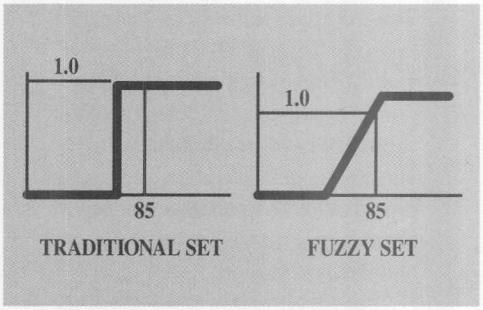

Figure 1.

By allowing degrees of membership to sets, fuzzy set theory allows knowledge to be represented in a more natural qualitative form. Furthermore, since fuzzy set theory clearly defines operations that can be performed on fuzzy sets, it allows the evaluation of implications or assertions. For example, the linguistic term "AND" is usually handled by evaluating a minimum of two fuzzy sets.

Continued on page 2

Continued from page 1

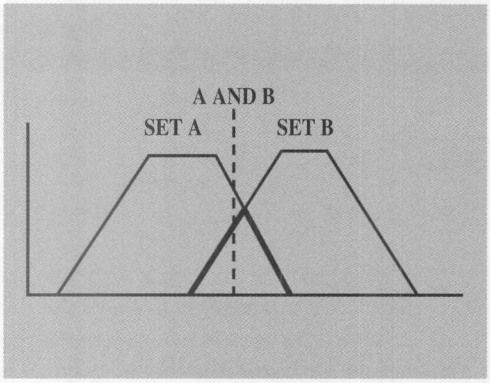

Figure 2.

Figure 2 shows this graphically. Similarly, the linguistic "OR" is handled by taking a maximum.

As an example of how assertions are evaluated, consider a rule such as: "IF speed is fast and distance is close THEN brake pressure is high." Assume that speed is fast with a membership of 0.75 and distance is close with a membership of 0.83, the brake pressure will be made high with a factor of 0.75.

Fuzzy logic allows control systems to be built which rely on domain knowledge of an expert to function rather than a complex mathematical model of a process. This "domain knowledge" usually takes the form of IF-THEN statements. These statements describe how the output(s) of the controller should behave given the state of various inputs. Generally the control process can be broken down into three stages: fuzzification, inference and defuzzification.

The fuzzification stage takes all of the inputs to the systems and determines their degree of membership to each relevant class. That is each input variable is mapped into all fuzzy sets associated with it.

Next, the inference stage evaluates each relevant rule in the system to determine its effect on the output of the controller. Usually, the minimum operator is used to determine the overall strength of the antecedents, as in the speed-distance example.

The defuzzification stage determines a crisp value for each output variable. The output is fed into the system to be controlled. There are many different ways to defuzzify the output such as extracting the center of gravity of the output from each rule, calculating the mean of the maximums from each rule, or a winner takes all approach.

The most important concept to note relating to fuzzy control system development is that the knowledge of how the controller should function is directly encoded into the system. It allows expert knowledge of controller behavior to be easily integrated into control strategy. It is not necessary to first develop a mathematical model of how the controlled process operates. This allows the control system designer to concentrate on how the process should be controlled.

Thus, fuzzy logic can be used as a tool to aid in the development of control systems. Also, there are now numerous software tools available from various companies that assist designers in developing fuzzy logic based control systems. *fuzzyTECH** from Inform Software Corporation is one such tool (see Common Ground Vol. 1, No. 6). This tool allows designers to enter membership functions and rules graphically, and then simulate the resulting controller. The tool will then automatically generate code for a target microcontroller.

The generated code consists of library function code, rule and membership function data and the fuzzy system code. The library code is fixed in size. The equation in Figure 3 gives the approximate amount of memory needed for the knowledge base. The amount of time needed to calculate the time required to evaluate all of the rules is left as an exercise for the reader.

For example a fuzzy controller with 9 rules of 2 input variables and 1 output variable with 5 terms each would use about 146 bytes of ROM. The total ROM required, including the needed library code and the "glue" code needed to set up the library calls is about 3000 bytes.

Many have argued that using fuzzy logic will yield better controllers than conventional techniques, but this is debatable. Fuzzy logic should be looked upon as a tool or technology that may be used to assist in the development of control systems. In this respect, fuzzy logic may allow better controllers to be designed more quickly than can be accomplished using conventional techniques.

$$mem = \sum_{i=1}^{nI} 8t_i + \sum_{j=1}^{no} 2t_j = \sum_{r=1}^{nR} (I_r + 2 * O_r + 2) + m$$

nI	=	Number of input variables
no	=	Number of output variables
nR	=	Number of rules
t_i	=	Number of terms for linguistic input variable i
t_j	=	Number of terms for linguistic output variable j
Ir	=	Number of input conditions for the rule r
Or	=	Number of output conditions for the rule r

Figure 3.

FUZZY LOGIC, CHANGING OUR VIEW

By Bert Hellenthal
Inform Software Corporation

There is nothing fuzzy about fuzzy logic. But there definitely is a naming problem. Would you buy a fuzzy auto focus on your 35mm camera?

Fuzzy logic is heavily used in automotive applications, an industry where the R&D departments are very advanced.

Some applications are:

Anti Locking Brake Systems (ABS)
Anti Skid Systems
Automatic Transmissions
Fuel Injection Systems
Exhaust Control
Active Suspension
Active Steering Support Systems
Crash Test Simulators
Production Scheduling
Air Conditioning Control

Problems and Solutions

What is fuzzy logic? Why do market studies predict a billion dollar market by 1998? Think of fuzzy logic as an enabling technology. You implement knowledge in the form of IF...THEN... rules and create solutions to your control application down to the system level. It is the next higher level of abstraction to solve real world problems. Developing applications with fuzzy logic is a design method more focused on solving the problem rather than encoding it. The way to develop the control algorithms has drastically changed. Knowledge and engineering heuristics as well as intuition can transform the application from closed loop to open loop decision support systems.

Will it replace conventional control engineering based on mathematical models? No, there are areas where nothing champions the traditional technology like a working calibrated Proportional Integral Differential (PID) controller. Most real-time applications are highly nonlinear and have time constraints. Thus they require a highly precise mathematical model to develop a traditional nonlinear control algorithm that tends to be expensive and difficult to document. By using this knowledge about the application and its problem, fuzzy logic enables a new working approach to design control algorithms.

Rules needed to describe rough and basic driving skills

IF the obstacle is far away	THEN	go fast
IF the obstacle is medium far away	THEN	go slow
IF the obstacle is close	THEN	stop
IF the obstruction is to the left	THEN	turn right
IF the obstruction is to the right	THEN	turn left
IF there is obstruction	THEN	go straight

Figure 1.

Automated vehicle control

One of the complex problem areas where a fuzzy logic solution was tested for feasibility was automated vehicle control. Looking at this problem from a conventional point of view, the task of modeling a real world car with precise mathematical models hasn't been solved. Traction, tire treads, suspension, transmission and road surface are a few variables in the system that must be considered.

Expressing the knowledge needed for basic driving skill takes about 9 rules (as shown in Figure 1). These rules are simple: IF an obstacle is close, THEN use the brakes or IF there is no obstacle ahead THEN drive straight.

To prove these simple rules work, a model car was built. To measure distance, 3 ultrasonic sensors were installed. Distance to the left, right and front was measured. A one horsepower electrical motor was used to power the car, rendering a power-to-weight-ratio of a Ferrari Testarossa. On dry surfaces the car reaches 20 mph in about 3.5 seconds with a top speed of 50 mph for 1 minute. The car was equipped with disk brakes and an individual differential suspension on each wheel.

The computer running the car was based on an Intel 12 MHz 80286 microprocessor. For more advanced testing and for research in neuro-fuzzy control, a 4 node transputer system was mounted on the back of the car. The system performance was 40MIPS/4.5 MFLOPS. Figure 2 shows a block diagram of the hardware.

The first fuzzy system, representing a rather primitive control strategy to run the car, was implemented by two engineers in 2.5 hours using on-line technology. The fuzzy system took the distances and speed as input data to control the steering and power (shown in Figure 3). The on-line technology enabled real-time remote cross debugging on a running target using software and a serial port (RS232C communication). Rules and membership functions of the fuzzy system could be dynamically changed on the target without halting the system. Consequently the fuzzy system could be analyzed and optimized on the fly.

Based on this experiment, a two step control strategy was implemented. Sensor data was interpreted to describe the car's situation. Then the appropriate action was determined to stabilize the car in skidding and sliding conditions automatically. This system describes a more complex strategy where rpm of each wheel are needed as input data, in order to control the steering, power

Continued on page 4

Continued from page 3

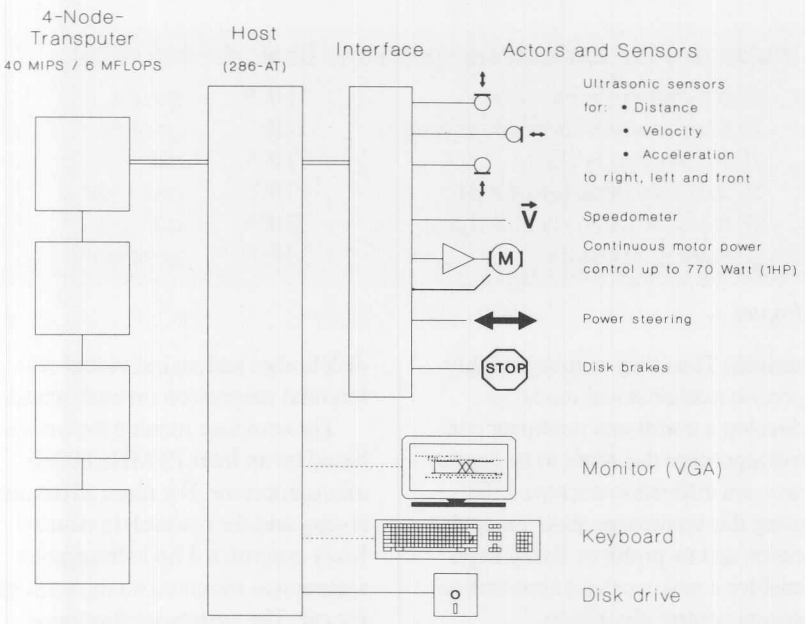

Figure 2. Hardware of the model car. Five processors are employed: Four T800 transputers are connected by high-speed serial links, each with a 4 MB individual memory. To load the control algorithm into the transputer from a 3½ disk drive and to drive the interfaces, an Intel 80286 processor is used.

and brakes[1]. (Figure 4 shows the structure of this system.)

Examination of this control concept indicates, automated driving and stabilizing with fuzzy logic is much more useful than other anti-skid implementations. Here advanced fuzzy logic methods together with an on-line approach enabled the design engineer to approach the problem from a much higher level of abstraction.

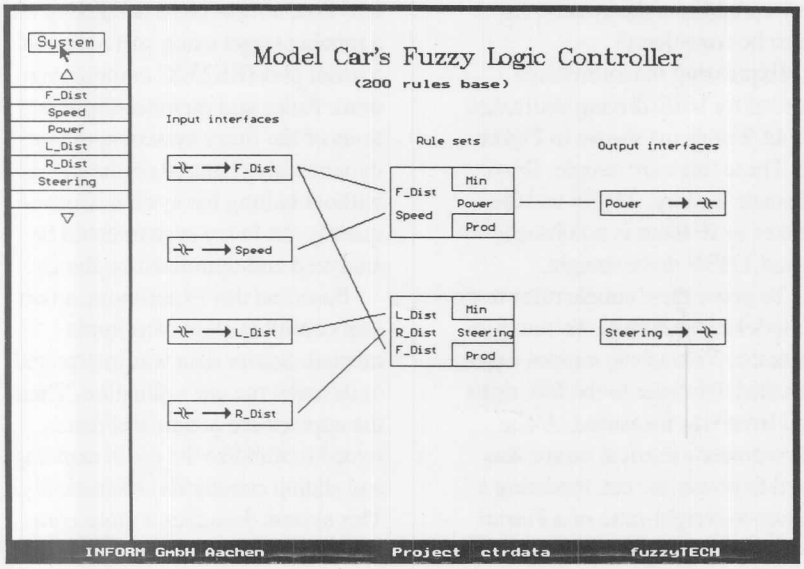

Figure 3. Structure of the simple control strategy with 200 rules (screenshot of the fuzzy*TECH shell).*

Considerations

People are fascinated by the concept of fuzzy logic and the new opportunities it presents. Design cycle development time can be cut drastically in many products. It also allows the upgrade of existing products by implementing more "knowledge" without additional hardware.

Using fuzzy logic technology is a new way to solve problems with code that is short and fast. To make development easier, fuzzy logic tools help with analysis, debugging and also compile to ANSI-C or proprietary microcontroller assembly code.

Stability of fuzzy systems

A commonly asked question is: "Can you prove the stability of a fuzzy system?" The answer is YES. It is the same as with any deterministic conventional nonlinear control system influenced by multiple parameters. All conventional control engineering methods can be used. There is no difference between a system developed using fuzzy logic or the conventional nonlinear control theory. Fuzzy logic doesn't release anyone from the proof of stability if it is needed. Stability must be proven on a mathematical model often highly complex, differential in nature and expensive.

Another way to test stability is extensive prototype testing on real world applications. This actually is the way most of today's nonlinear and multiparameter controllers are approved. In some cases a traditional proof of stability may still be necessary for legal, insurance and government approval. Again, all known tools and methods from traditional nonlinear control theory are applicable to fuzzy systems.

Continued on page 8

AUTOMOTIVE MCS®51 CONTROLLER ICC SPECIFICATION NOTICE

David Elting
Applications Engineer, Automotive Applications, Intel Corporation

There have been questions regarding Icc specifications for Automotive MCS 51 controllers. The intent of this article is to clarify Icc specifications. The Icc specifications listed in this document will appear in the next Automotive Handbook planned for release in January 1994. Please disregard all previous Icc specification notices.

Data sheets this document applies to

The Icc maximum specification applies to the following data sheets:

1. *87C51FA/87C51FB/87C51FC/87C51FC-20 CHMOS Single-Chip Microcontroller*
2. *87C54/87C54-20 CHMOS Single-Chip 8-Bit Microcontroller*
3. *80C51FA/83C51FA CHMOS Single-Chip Microcontroller*
4. *80C31BH/80C51BH/87C51 CHMOS Single-Chip Microcontroller*
5. *8031AH/8051AH/8032AH/8052AH NMOS Single-Chip Microcontrollers*

Please note the following changes from the 1992 Automotive Handbook.

The new 87C51FA/87C51FB/87C51FC/87C51FC-20 data sheet combines the 87C51FA, 87C51FB/87C51FC, and the new 87C51FC-20, a 20 MHz 87C51FC, data sheets. The new 87C54/87C54-20 data sheet replaces the 87C54 data sheet and incorporates the new 87C54-20, a 20 MHz 87C54. The 20 MHz 87C51FC and 87C54 are new offerings for the 8-bit FX core devices.

Updated Icc specifications by product

In the following section an asterisk indicates a change in Icc specification.

1. 87C51FA/87C51FB/87C51FC/87C51FC-20:

Symbol	Parameter	Max	Unit	Test Conditions
Icc	Power Supply Current:			(see note)
	Running at 16/20 MHz	35/40	mA**	
	Idle mode at 16/20 MHz	12/14	mA**	
	Power-down mode	100	uA	

Note: Icc is measured with all output pins disconnected. Max. Icc (for Active and Idle modes) ratings determined using Vcc=6V.

The new Icc max equation is:

Active mode

Icc max = (1.25 x Osc Freq) + 15**

2. 87C54/87C54-20:

Symbol	Parameter	Max	Unit	Test Conditions
Icc	Power Supply Current:			(see note)
	Running at 16/20 MHz	28/33	mA**	
	Idle mode at 16/20 MHz	12/14	mA**	
	Power-down mode	100	uA	

Note: Icc is measured with all output pins disconnected. Max. Icc (for Active and Idle modes) ratings determined using Vcc=6V.

The new Icc max equation is:

Active mode

Icc max = (1.25 x Osc Freq) + 8**

3. 80C51FA/83C51FA:

Symbol	Parameter	Max	Unit	Test Conditions
Icc	Power Supply Current:			(see note)
	Running at 12 MHz	40	mA	
	Idle mode at 12 MHz	15	mA	
	Power-down mode	150	uA	

Note: Icc is measured with all output pins disconnected. Max. Icc (for Active and Idle modes) ratings determined using Vcc=6V.

The new Icc max equation is now:

Active mode

Icc max = (3 x Osc Freq) + 4**

4. 80C31BH/80C51BH/87C51:

Symbol	Parameter	Max	Unit	Test Conditions
Icc	Power Supply Current:			(see note)
	Running at 12 MHz			
	80Cx1BH/87C51	20/25	mA**	
	Idle mode at 12 MHz			
	80Cx1BH/87C51	5/6	mA**	
	Power-down mode			
	80Cx1BH/87C51	75/100	uA**	

Note: Icc is measured with all output pins disconnected. Max. Icc (for Active and Idle modes) ratings determined using Vcc=6V.

The new Icc max equation is:

Active mode

Icc max = (0.94 x Osc Freq) + 13.71**

**New specifications

VERIFYING ROM CODE ON LOCKED 8XC196KX/JX FAMILY DEVICES

Dave Boehmer
Applications Engineer, Automotive Applications, Intel Corporation

Print files are being distributed on diskette in this issue of COMMON GROUND. The schematic that accompanies this article can be printed by entering the following command from the DOS prompt:

"COPY ROMVER.PRT LPTx"

LPTx can be LPT1, 2 or 3 depending upon where your print device is located. The package will include the schematic in both a Postscript print file format (ROMVER.PRT) and Protel Advanced Schematic format (ROMVER.SCH).*

Further information on programming and verifying 8XC196Kx Family microcontrollers can be found in the 8XC196Kx Family User's Guide, literature order #272258-001.

The ROM Verification Board provides a simple, stand-alone method for verifying ROM codes of 8XC196Kx/Jx devices after LOC bits have been enabled. Although this article details verification of 52-lead 8XC196Jx devices, this method can easily be adapted for use with 68-lead 8XC196Kx devices. To use the ROM Verification Board, the user must be able to program two 27C512 EPROMs with the correct Security Key, ROM Code, and PPW. The user must also supply ground, 5.0 volts, and 12.5 volts to the board.

Normal auto programming mode operation

The ROM Verification Board works by initiating the Auto Programming mode of the 8XC196JR/JQ device. The auto Programming mode in internal Test ROM is entered by applying 12.5 volts on the EA# pin and a PMODE value of "0Ch" on the upper nibble of Port 0 during RESET. On the ROM Verification Board, the green LED illuminates after a system reset and remains on until a programming error is encountered.

After entering Auto Programming mode, the CCB LOC bits are checked to determine if the user has locked the device against unauthorized reads or writes. If the LOC bits are set, the external security key is checked against the internal 16-byte security key programmed into the JR/JQ at locations 2020h to 202Fh. If the security keys do not match exactly, the device enters an endless loop and prevents any unauthorized reads or writes of the device. If the security keys do match, then Auto Programming mode execution is continued.

After security key verification, the PPW (Programming Pulse Width) is fetched from external memory (location 0014h). Taking the address decoding scheme of the ROM Verification board into account, the PPW is actually fetched from external EPROM location 200Ah, as indicated in the table on the next page.

After the PPW is fetched, P2.7 (PACT#) is cleared to indicate that programming has begun. The yellow LED on the ROM Verification Board lights at this point and remains lit until all locations are programmed and verified.

Auto Programming mode begins by fetching the first word of data to be programmed from external memory. It then checks to see if the location should be blank (0FFFFh). If the word is all ones, that location is skipped and the next word to be programmed is fetched from external memory. When a word is fetched that is not blank, the respective internal EPROM location is programmed with the fetched data.

After a location is programmed, it is then read and verified against the data fetched from the external location. If it doesn't verify, P2.0 (PVER) is pulled low for the rest of the programming sequence, indicating an error. On the ROM Verification Board, the red LED lights (and the green LED goes out) to indicate that a verification error has occurred.

After all locations are programmed and verified, P2.7 (PACT#) goes high, indicating that the Auto Programming routine has finished, and the yellow LED on the board goes out. If no errors were encountered, the green LED remains lit. If a verification error occurred, the red LED remains lit.

ROM Verify Operation

The ROM Verification Board differs slightly from what would normally be implemented for an Auto Programming board. The difference is Vpp is tied to 5 volts instead of 12.5 volts. The device goes through the steps of programming itself with the external data but, due to the lack of adequate programming voltage, is not programmed. This in effect means only the verification part of the Auto Programming mode takes place.

Limitations

Two very important things should be considered when using the ROM Verification board to verify locked ROM contents. The first, and most important, is blank locations are not verified. If an external memory location is blank (0FFFFh), this method does not verify the blank condition of corresponding ROM locations.

The second thing to note is if th security keys do not match, the green LED will stay lit after the yellow LED goes out. This could be misinterpreted as ROM content being verified. If the security keys do not match, the yellow LED lights very briefly and goes out

almost immediately after the RESET button is released. It is assumed that the user will supply the correct external security key to allow the device to enter Auto Programming Mode. However, if this causes concern, the circuit could be modified to light another LED until the PPW fetch. This LED would signify the security key did not match.

Memory Map

The table describes the addresses resulting from the address decoding scheme of the board.

As indicated in the table, the user must locate the Security Key (exactly as programmed into the JR/JQ at locations 2020h to 202Fh) at external EPROM locations E010h to E017h. The PPW must be located at external locations 200Ah (80FFh is an acceptable value). The JR/JQ's internal ROM code to be verified must be located at 6000h to 7FFFh. 52-lead devices default to 16-bit mode for the Auto Programming Mode, therefore, the code to be verified against must be split (high byte, low byte) between two EPROMs in order to work properly.

JR/JQ generates:	Address at EPROM:	Location contains:
C020h	E010h	Beginning of Security Key
•	•	•
•	•	•
C02Eh	E017h	End of Security Key
0014h	200Ah	PPW
4000h	6000h	Beginning of data to be verified
4002h	6001h	•
•	•	•
•	•	•
7FFEh	7FFFh	End of data to be verified**

*** This range may need to be adjusted, depending upon the ROM length of the device being verified. Please refer to the 8XC196Kx Family User's Manual for details.*

Using the ROM verification board

Verifying locked ROM contents with the ROM Verification Board requires a simple procedure:

1. Program the two 27C512s (high byte, low byte):
   - Security Key at E010h to E017h (must match internal Security Key).
   - PPW at 200Ah (80FFh is an acceptable PPW).
   - Code that the JR/JQ is to be verified against at 6000h to 7FFFh.
2. Place the two 27C512s into their respective ZIF sockets on the board.
3. Place the JR/JQ device to be verified into the socket provided on the board.
4. Apply ground, 5.0 volts, and 12.5 volts to the board simultaneously. **Warning:** Applying 12.5 volts to the board before applying 5.0 volts can damage the JR/JQ device.
5. Press the RESET button. The green LED lights after a system reset.
6. The yellow LED lights, indicating that the Auto Programming sequence has begun.
7. After the sequence has finished, the yellow LED goes out and either the red or the green LED remains lit. The red LED indicates that a verification error has occurred. The green LED indicates that the programmed contents verified correctly. The only exception to this would be if an incorrect external security key was supplied.

Conclusion

This article was written to demonstrate the benefits of using fuzzy logic. It is useful to take a closer look at how fuzzy logic can be beneficial to your business.

If you are interested in fuzzy logic technology, or are skeptical and would like to see applications in detail, Intel Corporation offers fuzzyBUILDER Kits as a total hardware and software application solution starting at $396.00. FaxBack* information can be received by calling 800-628-2283 and requesting document #2174.

If you would like to see more automotive applications using fuzzy logic, such as ABS, ASR, engine control, etc. or would like an introduction for engineers to the theory of fuzzy logic, please let us know. Send a fax to Charly Gullett at Intel Automotive Marketing (602) 554-8820.

It is worthwhile to consider fuzzy logic, as indicated by the appearance of fuzzy applications on the market and market studies that predict an eight billion dollar market by 1998.

(For a more complete paper on automated vehicle control call 800-929-2815.)

[1] von Altrock, Krause, Zimmerman "Advanced fuzzy logic control of a model car in extreme situations"; Fuzzy Sets and Systems 48 (1992) 41-52

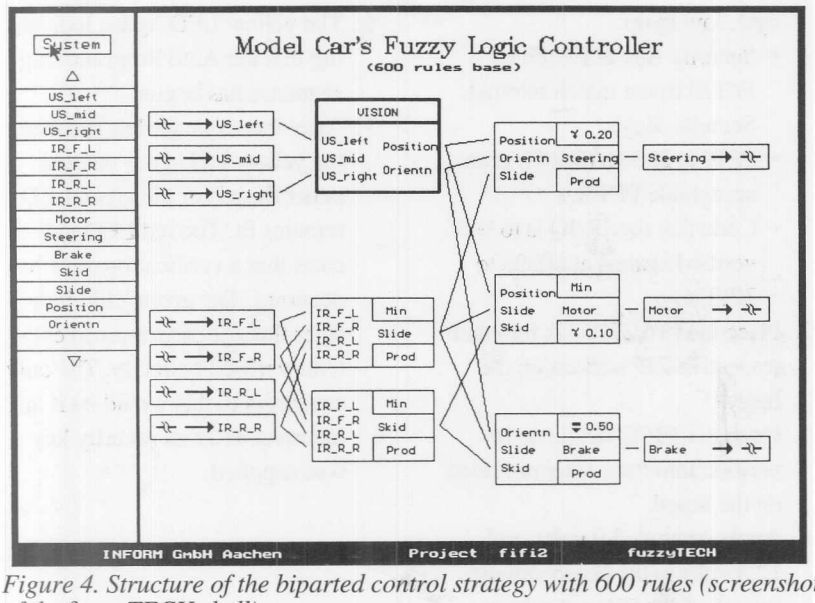

Figure 4. Structure of the biparted control strategy with 600 rules (screenshot of the fuzzy*TECH shell).*

EDITOR'S COLUMN

My Grandfather used to say things never change. Some things, of course, have changed. For instance, our journal being typeset instead of copied and brought to you on recycled paper. To my grandfathers credit, some things have not changed. You will find here, as before, relevant technical information, design tips and thoughtful articles, a diskette is also included. Two files are schematics to accompany David Boehmer's fine article on ROM verification. To print, follow David's instructions at the beginning of the article.

You will also find a file called SETUP.EXE on the disc. This is a self-installing Microsoft Windows program. Execution will create a Windows group called COMMON GROUND and install a demonstration program icon for FUZZY LOGIC. Be sure to pull down the "About..." menu to gain valuable insight to the rule base used in this demonstration.

COMMON GROUND will continue, somewhat changed, but pretty much the same. I sincerely hope you like the new combination.

Editor

Printed on Recycled Paper

Common Ground Newsletter
Volume 2, Number 1
Editor: Charly Gullett (602) 554-8420
Production Editor: Allegra Sinclair